Voľba šéfa v úplnom grafe

Majme N procesov zapojených do úplného grafu. Každý z nich má jednoznačné ID a vie rozlíšiť medzi svojimi linkami. Okrem rôznych ID sú procesy identické a pracujú asynchrónne. Na začiatku je zobudených niekoľko (aspoň jeden) procesov. Keď proces dostane správu, zobudí sa. Uvažujeme obojsmerné full-duplex FIFO linky. Každá poslaná správa v konečnom čase príde do cieľa. Algoritmus je zapísaný tak, že pri zobudení (spontánnom alebo ľubovoľnou správou) sa najprv vykoná sekcia Init a potom sa vykonáva kód špecifikovaný v sekcii Code. Pri príchode správy nastane "prerušenie" a spracuje sa príslušný odsek On receipt; potom sa pokračuje vo vykonávaní v sekcii Code alebo (pomocou príkazu **goto**) v inej sekcii. Pre určenie časovej zložitosti predpokladáme, že doručenie správy po linke trvá maximálne 1 časovú jednotku a všetky interné výpočty trvajú 0 časových jednotiek.

1 Algoritmus $\mathcal{A}$

Naivný algoritmus, v ktorom každý vrchol pošle všetkým ostatným správu, že chce byť šéf a čaká, či mu to dovolia. Iba procesu s najvyšším ID všetky ostatné procesy schvália kandidatúru. Algoritmus pre proces vo vrchole v:

const: N : integer
ID : integer
$Neigh$: [1...N-1] link
var: $leader$: boolean
$count$: integer
i : integer

Init:

$count := 0$
$leader :=$ **false**

Code:

for $i = 1$ **to** $N - 1$ **do**
  send $\langle$**elect**$, ID\rangle$ **to** $Neigh[i]$
while $count < N - 1$ **wait**
for $i = 1$ **to** $N - 1$ **do**
  send $\langle$**leader**$, ID\rangle$ **to** $Neigh[i]$
$leader :=$ **true**

On receipt $\langle$**elect**$, id_i\rangle$ from $Neigh[i]$:

if $id_i > ID$ **send** $\langle$**accept**$\rangle$ **to** $Neigh[i]$

On receipt $\langle$**accept**$\rangle$ from $Neigh[i]$:

$count++$

On receipt $\langle$**leader**$, id_i\rangle$ from $Neigh[i]$:

Skonči algoritmus

Aby sme dokázali správnosť, treba ukázať, že v ľubovoľnom výpočte sa práve jeden proces stane šéfom. Predpokladajme sporom, že sa ani jeden nestane šéfom. Každý zobudený proces pošle správy všetkým ostatným procesom, teda aj procesu s najvyšším ID a tým ho zobudí (ak už nebol zobudený). Proces s najvyšším ID pošle **elect** všetkým ostatným procesom a dostane naspäť všetky **accept** správy (lebo žiaden z procesov neskončil – to by musel dostať správu **leader** alebo sa sám vyhlásiť za šéfa) a vyhlási sa za šéfa. Ďalej predpokladajme, že sa za šéfa vyhlásia aspoň

dvaja. V tom prípade musel ten s menším ID niekedy dostať **accept** správu od toho s väčším ID, čo je spor.

Po každej hrane sa posielajú v jednom smere maximálne 3 správy, teda počet správ je $O(n^2)$. Proces s maximálnym ID dostane za 2 časové jednotky naspäť všetky **accept** správy, celková zložitosť je teda $O(1)$.

2 Algoritmus $\mathcal{B}$

Mierne rafinovanejší algoritmus, v ktorom sa každý proces snaží získavať povolenia sekvenčne. Pri porovnávaní sa neberie do úvahy iba ID, ale aj "level" (t.j. počet porazených procesov). Proces posiela správy **capture** a vždy čaká na odpoveď **accept**. Tá však príde iba vtedy, keď vyhral. Odpoveď na správu **capture** závisí od toho, či už proces bol raz porazený. Ak nie, rozhoduje veľkosť $[Level, id]$ (lexikograficky). Ak áno, porovnanie nerobí on, ale jeho "rodič" (t.j. proces, ktorý ho poslednýkrát "zajal"). Algoritmus pre proces vo vrchole v:

const: N : integer
ID : integer
$Neigh$: [1...N-1] link
var: $leader$: boolean
$state$: {active, captured, killed}
$level$: integer
$parent$: link
msg : {victory, defeat}
i : integer

Init:

$state :=$ active
$level := 0$
$leader :=$ **false**

Code:

for $i = 1$ **to** $N - 1$ **do**
  send $\langle$**capture**, $[level, ID]\rangle$ **to** $Neigh[i]$
  receive $\langle$**accept**$\rangle$ **from** $Neigh[i]$
  $level++$
$leader :=$ **true**
for $i = 1$ **to** $N - 1$ **do**
  send $\langle$**leader**, $ID\rangle$ **to** $Neigh[i]$

Dead:

loop forever

On receipt $\langle$**capture**, $[level_i, id_i]\rangle$ from $Neigh[i]$:

if $state \in$ {active, killled} **and** $[level_i, id_i] > [level, ID]$
  $state :=$ captured
  $parent := Neigh[i]$
  send $\langle$**accept**$\rangle$ **to** $parent$
  goto Dead
else if $state =$ captured
  send $\langle$**help**, $[level_i, id_i]\rangle$ **to** $parent$
  receive msg **from** $parent$
  if $msg =$ defeat
    send $\langle$**accept**$\rangle$ **to** $Neigh[i]$
    $parent := Neigh[i]$

On receipt $\langle$**help**, $[level_i, id_i]\rangle$ from $Neigh[i]$:

if $[level_i, id_i] < [level, ID]$
  send $\langle$victory$\rangle$ **to** $Neigh[i]$
else
  send $\langle$defeat$\rangle$ **to** $Neigh[i]$
  if $state =$ active
    $state :=$ killed
    goto Dead

On receipt $\langle$**leader**, $id_i\rangle$ from $Neigh[i]$:

Skonči algoritmus

Opäť najprv ukážeme, že v ľubovoľnom výpočte sa práve jeden proces stane šéfom. Dokážeme takúto lemu:

Lema 1 *V ľubovoľnom výpočte existuje pre každý level $l = 0, ..., N - 1$ aspoň jeden proces, ktorý bol počas výpočtu na leveli l.*

Dôkaz: Pre $l = 0, 1$ je tvrdenie triviálne. Ďalej postupujme sporom. Zoberme maximálny level l taký, že v priebehu výpočtu bol nejaký proces na leveli l ale žiaden proces nebol na leveli $l + 1$.

Zoberme proces v s maximálnym ID i spomedzi procesov na leveli l. Keďže v nepostúpil o level, nastala jedna z troch možností: niekto mu zmenil stav na "captured" (resp. "killed") alebo v poslal **capture** a nedostal odpoveď. Zmeniť stav je možné iba správami **capture** alebo **help**, ak obsahujú väčší (lexikograficky) proces. To je ale spor s tým, že v je (v tomto usporiadaní) maximálny. Takže v poslal správu **capture** povedzme procesu v'. Ak v' bol aktívny alebo zabitý, pošle okamžite odpoveď **accept** (lebo je lexikograficky menší). Ak v' je zajatý, opýta sa svojho rodiča; ten je však podľa predpokladu tiež lexikograficky menší ako v, takže v postúpi o level. □

Keďže v priebehu výpočtu existoval (apoň jeden) proces s levelom $N-1$, aspoň jeden proces sa vyhlási za šéfa. Teraz dokážeme ešte jednu lemu:

Lema 2 *Nech v je aktívny proces (state =* active*) s levelom l. Potom existuje l zajatých procesov ktoré patria v (t.j. ich premenná parent ukazuje na v).*

Dôkaz: Proces postúpi o level iba vtedy, keď dostal **accept**. Poslanie správy **accept** je vždy doprevádzané zmenou stavu na "captured" a nastavením *parent*. Takže pri každom postúpení o level sa počet zajatých procesov zväčší. Tento počet sa môže zmenšiť iba ak si niektorý z nich nastaví *parent* inam. Z kódu ale vyplýva, že potom v prejde do stavu "killed" a nie je viac aktívny. □

Z uvedenej lemy vyplýva, že na leveli $N-1$ môže byť počas celého výpočtu najviac jeden proces. (dôkaz: zoberme prvý proces, ktorý sa dostane na level $N-1$; všetky zvyšné procesy sú v stave "captured" a na menšom leveli; keďže v stave "captured" vždy vykonávajú sekciu Dead, žiaden z nich už nikdy viac nepostúpi o level). Takže za šéfa sa vyhlási práve jeden proces.

Ideme ukazovať počet správ. Zrejme ak nejaký proces postúpi o level, spotrebuje na to konštantný počet správ (**capture**, príp. **help**+defeat, **accept**). Ak nejaký proces pošle **capture** a napriek tomu nepostúpi o level (t.j. nedostane **accept**) prestane byť aktívny a vymení sa tiež konštantý počet správ (**capture**, príp. **help**+victory). Neaktívne procesy neposielajú spontánne správy. Z toho vyplýva, že proces na každom leveli spôsobí poslanie konštantného počtu správ. Teraz dokážeme takúto lemu:

Lema 3 *V ľubovoľnom výpočte je najviac $N/(l+1)$ procesov, ktoré niekedy dosiahli level l.*

Dôkaz: Pre každý proces v, ktorý dosiahol niekedy level l označme C_v tie procesy, ktoré patrili v (t.j. boli zajaté a *parent* mali nastavené na v) vtedy, keď dosiahol level l. Zrejme $|C_v| = l$. Ukážeme, že množiny C_v sú po dvoch disjunktné. Dokážeme to indukciou na čas, kedy proces dosiahol level l. Prvý proces v_1 dosiahol level l a mal množinu C_{v_1}. Predpokladajme, že procesy $v_1, ..., v_k$ dosiahli level l a množiny $C_{v_1}, ..., C_{v_k}$ sú navzájom disjunktné. Majme proces v_{k+1}, ktorý je na leveli $l-1$ a ide postúpiť na level l. Zrejme mu teraz patrí $l-1$ procesov, ktoré nepatria do $C_{v_1} \cup ... \cup C_{v_k}$ (v_{k+1} nemohol získať žiadny proces, ktorý už raz mal rodiča na leveli l). Takisto na postup na level l nemôže v_{k+1} použiť proces z $C_{v_1} \cup ... \cup C_{v_k}$ (to by ho zabilo), takže musí pridať iný proces. □

Z uvedeného vyplýva, že v ľubovoľnom výpočte sa maximálne $\sum_{l=1}^{N-1} \frac{N}{l+1}$ krát posunie nejaký proces o level. Keďže $\sum_{l=1}^{N-1} \frac{N}{l+1} = N(\mathbf{H}_N - 1) \approx N \log N$, v celom algoritme sa použije $O(N \log N)$ správ.

Na určenie časovej zložitosti najprv ukážeme, že proces, ktorý bude zvolený za šéfa sa zobudí najneskôr $O(N)$ časových jednotiek po začiatku. Stačí ukázať, že najneskôr $O(N)$ časových jednotiek po začiatku niektorý proces postúpi aspoň na level 1 (potom už nemôže byť zvolený šéf s levelom 0). Ako dlho môže trvať situácia, že všetky procesy sú na leveli 0? Zrejme tak dlho, pokiaľ všetky **capture** správy idú do procesov s väčším ID. To zrejme môže byť najviac $N-1$ (N správ už nutne vytvorí cyklus). Takže najneskôr po $O(N)$ časových jednotkách je zobudený šéf. Šéf vždy v konštantnom čase postúpi o level, takže celková zložitosť je $O(N)$.

Poznámka: Časová zložitosť $O(N)$ ostane zachovaná, aj keď použijeme model kedy po jednej linke v jednom smere môže ísť vždy iba jedna správa. V tom prípade ale neplatí, že šéf zajme každý vrchol v konštantnom čase.